UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/025,270	12/18/2001	Donald Robert Syme	180610.01	5667

22971 7590 03/05/2007
MICROSOFT CORPORATION
ONE MICROSOFT WAY
REDMOND, WA 98052-6399

EXAMINER	
YIGDALL, MICHAEL J	
ART UNIT	**PAPER NUMBER**
2192	

SHORTENED STATUTORY PERIOD OF RESPONSE	NOTIFICATION DATE	DELIVERY MODE
3 MONTHS	03/05/2007	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Notice of this Office communication was sent electronically on the above-indicated "Notification Date" and has a shortened statutory period for reply of 3 MONTHS from 03/05/2007.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

jranck@microsoft.com
roks@microsoft.com
ntovar@microsoft.com

Office Action Summary	**Application No.** 10/025,270	**Applicant(s)** SYME ET AL.
	Examiner Michael J. Yigdall	**Art Unit** 2192

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *29 November 2006*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-14,25 and 26* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-14,25 and 26* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.	6) ☐ Other: _____.

DETAILED ACTION

1. This Office action is responsive to the appeal brief filed on November 29, 2006. It has been determined that the application is not in form for appeal because additional rejections under 35 U.S.C. 101 are necessary, as set forth below. Claims 1-14, 25 and 26 are pending.

2. Applicant indicates that a notice of appeal was filed on September 9, 2006 in response to an advisory action and a final Office action (brief, page 4, top). It is noted, however, that the notice of appeal was filed on September 29, 2006, and that it was in response to a non-final Office action mailed on March 31, 2006. No final Office actions or advisory actions were mailed subsequent to the action of March 31, 2006.

Response to Arguments

3. Applicant's arguments have been fully considered but they are not persuasive.

Applicant contends that "Viroli discusses a possible implementation of a parametric polymorphism system but does not disclose the internal mechanisms by which such an implementation may be completed" and further opines that "the parametric polymorphism system of Viroli is not specific and the specific internal structures of Viroli can not be extrapolated from sample code for use in the system and a general discussion of the functioning of the system" (brief, page 13).

However, this argument is not compelling. Certainly Viroli's teachings enable any person skilled in the art to make and use the parametric polymorphism system. Furthermore, as Applicant notes, the reference must show the invention only in as complete detail as is contained in the claim (brief, page 7). That is to say, Viroli must show only the "internal mechanisms" and "specific internal structures" that are specifically recited in Applicant's claims. Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

Applicant contends that Viroli does not disclose "a typing context, a typing context handle, or the action of identifying a typing context handle associated with the typing context" (brief, page 14).

First, however, Applicant is respectfully reminded that the claims are given the broadest reasonable interpretation. A "typing context" is interpreted as an abstract state or "context" that is in some manner related to "typing." Even according to Applicant's characterization of the reference (brief, page 11), Viroli's parametric polymorphism system is thus directed to such "typing contexts."

Viroli teaches the action of "identifying a typing context handle associated with the typing context" in terms of retrieving a type descriptor associated with the typing context. Viroli's "$TDManager" class, the type descriptors manager, enables this action. Specifically, the type descriptors manager provides a hash table of all type descriptors; retrieving a type descriptor from the hash table involves identifying a unique identifier or a "typing context handle" (see, for example, page 11, section 4.1, third paragraph). Indeed, the "handle" references a type descriptor (i.e., Viroli's "$TD" class), which is a "typing context data structure" as recited in the claims (see, for example, page 12, section 4.2, first paragraph). Notwithstanding Applicant's apparent contention that the system of Viroli does not actually include a hash table (brief, page 14), the hash table is the data structure from which the type descriptor or "typing context data structure" is retrieved.

Applicant contends that Viroli does not disclose "typing context data" or the action of "computing typing context data" (brief, page 15).

However, Viroli's type descriptors comprise "typing context data" in the sense that each one includes "the Class representation of the current type, the type variable instantiation, the friend types descriptor and the father type descriptor" (see, for example, page 12, section 4.2, first paragraph). Viroli "computes" the type descriptors in terms of creating and registering them (see, for example, page 12, section 4.2, fourth paragraph). While Applicant states that Viroli rejects the translation discussed in section 3, Viroli does implement a translation "based on the same mechanism" (brief, page 15). The translation effects the creation and registration of type descriptors (see, for example, page 12, section 4.2, second paragraph), which is to say the "computation of typing context data."

Furthermore, the examiner notes Applicant's admission that "instantiating parametric types" is a function of every programming language (brief, page 15). An instantiation of a parametric type is a "typing-context-relevant-code-point" as recited in the claims.

Applicant contends that Viroli uses "translated objects" rather than an "exact type" and that a translated object "is translated and not exact" (brief, page 15).

However, the examiner does not agree with Applicant's characterizations. "Translated" and "exact" are not mutually exclusive. In Viroli, the translated objects use exact types. Viroli illustrates the translation in Figure 10 (page 14) as compared to Figure 1 (page 4). Consider the object "c1" for example, whose declaration is translated to "Cell c1=new Cell($t[1],null)." The field "$t[1]" describes the exact type of the object "c1" because "$t[1]" represents the type descriptor for "Cell<Integer>." This is consistent with Applicant's *Discussion of Parametric Polymorphism* (brief, page 7), which states:

> Consider now a "generic" type that may be defined in the "Add" method of the base class. Such a generic type may indicate that a type may be any legal type. For example, the Add method may now be defined on the base class as "Add(<T> a, <T> b)". When an inherited class inherits from the base class and defines the "Add" method, the inherited class may define the type that "<T>" represents. For example, the inherited class may define <T> to represent and integer, a string, or the like. When an object is instantiated from the inherited class, the generic type <T> is replaced with the specific type. (Remarks, pages 10-11.)

Indeed, Viroli provides a similar discussion:

> Class Cell<T> is a simple wrapper for the type T, it is a parametric (or generic) class which abstracts from the type T, which is called its type variable (or its type parameter). ... As one may notice clients do not use parametric classes before having assigned actual types to all their type parameters. The type one obtains with such a binding operation, is called an instantiation of the parametric type. So for example Client uses the type Cell<Integer>, which is an instantiation for the class Cell<T> in which type variable T is instantiated to the type Integer. (Page 4, section 2, second paragraph).

The examiner agrees with Applicant that in the above example, "Cell<Integer>" is "an object instantiated from the 'Cell' class with an 'Integer' substituted for the generic open-type parameter" and that "an 'Integer' is an exact type" (brief, page 16). However, the examiner does not agree with Applicant's contention that "Cell<Integer>" does not describe an exact type (brief, page 16). The class "Cell" is, in fact, a type. While "Cell<T>" would describe a generic type, "Cell<Integer>" describes an exact type. The plain language of the claims does not distinguish over the teachings of Viroli.

Applicant states that the Office "attempts to draw an equivalency between 'a field in the typing context data structure' and a 'type descriptor' of Viroli and an identical equivalency between 'typing context data' and a 'type descriptor' of Viroli" (brief, page 17).

More precisely, to the extent recited in the claims, Viroli's type descriptor is equivalent to a "typing context data structure," and as noted above, the type descriptor includes fields that store "typing context data."

In response to Applicant's arguments directed to claims 25 and 26, it is noted that the subject matter recited in claims 25 and 26 corresponds to that of claim 1. Accordingly, claims 25 and 26 are rejected for the same reasons as set forth for claim 1.

While Applicant makes general statements that Applicant "disagrees [that] the execution engine recited in claim 25 corresponds to the computer program product of claim 1" (brief, page 17), and likewise "disagrees [that] the [method] recited in claim 26 corresponds to the computer program product of claim 1" (brief, page 18), the examiner notes that Applicant fails to describe with specificity any reasons why the subject matter should not correspond. Moreover, the examiner notes that in Applicant's *Summary of Claimed Subject Matter* (brief, page 4), Applicant relies on identical references to the specification to provide support for each of independent claims 1, 25 and 26.

In response to Applicant's arguments directed to the rejection of claims 1-14 under 35 U.S.C. 101 (brief, page 21), the examiner notes that any judgment on the *Interim Guidelines for Examination of Patent Applications for Patent Subject Matter Eligibility* (1300 OG 142) is fully outside the examiner's purview.

Claim Rejections - 35 USC § 101

4. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

5. Claims 1-14, 25 and 26 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Claims 1-14 are directed to a computer program product encoding a computer program. However, Applicant defines "computer program product" to include "a computer data signal embodied in a carrier wave" (specification, page 4, first paragraph). Signals and carrier waves do not fall within any class of statutory subject matter, and thus claims 1-14 are not limited to statutory subject matter. See *Interim Guidelines for Examination of Patent Applications for Patent Subject Matter Eligibility* (1300 OG 142), Annex IV, section (c).

Furthermore, claim 1 does not provide a practical application of the encoded computer program. As recited, the final result of the claimed subject matter is "recording the typing context data in the field of the typing context data structure." However, merely recording the typing context data in a data structure does not amount to useful, concrete and tangible result. Specifically, the claim does not recite any further use of the recorded typing context data that would permit its usefulness to be realized. In other words, the claimed subject matter does not produce a useful result because the usefulness of the recorded typing context data is never realized. Accordingly, claim 1 lacks a practical application and is therefore directed to non-statutory subject matter. See MPEP § 2106.

Dependent claims 2 and 4-14 do not remedy claim 1. Dependent claim 3, however, indicates that the recorded typing context data is used to create an instance of a class, which does appear to amount to a useful, concrete and tangible result.

Claim 25 is directed to an execution engine. However, the claim does not provide a practical application of the execution engine. As recited, the final result of the claimed subject matter is "recording the typing context data in the field of the typing context data structure." Merely recording the typing context data in a data structure does not amount to useful, concrete and tangible result. Specifically, the claim does not recite any further use of the recorded typing context data that would permit its usefulness to be realized. In other words, the claimed subject matter does not produce a useful result because the usefulness of the recorded typing context data is never realized. Accordingly, claim 25 lacks a practical application and is therefore directed to non-statutory subject matter. See MPEP § 2106.

Claim 26 is directed to a method. However, the claim does not provide a practical application of the method. As recited, the final result of the claimed subject matter is "recording the typing context data in the field of the typing context data structure." Merely recording the typing context data in a data structure does not amount to useful, concrete and tangible result. Specifically, the claim does not recite any further use of the recorded typing context data that would permit its usefulness to be realized. In other words, the claimed subject matter does not produce a useful result because the usefulness of the recorded typing context data is never realized. Accordingly, claim 26 lacks a practical application and is therefore directed to non-statutory subject matter. See MPEP § 2106.

Claim Rejections - 35 USC § 102

6. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

7. Claims 1-5, 7-9, 25 and 26 are rejected under 35 U.S.C. 102(b) as being anticipated by Viroli et al., *Parametric Polymorphism in Java through the Homogeneous Translation LM: Gathering Type Descriptors at Load-Time* (art of record, "Viroli").

With respect to claim 1 (previously presented), Viroli discloses a computer program product encoding a computer program for executing on a computer system a computer process for dynamically generating typing context data (see, for example, the abstract, and page 12, section 4.2, second paragraph, which further shows a translation that enables the dynamic generation of type descriptors, and see, for example, page 13, Figure 9, which shows a type descriptor as defined in class "$TD") associated with a typing-context-relevant-code-point being executed within a typing context in a dynamic execution environment (see, for example, page 11, section 4.1, first paragraph, which shows the instantiation of parametric types, i.e. the execution of typing-context-relevant-code-points), the computer process comprising:

(a) encountering the typing-context-relevant-code-point in the typing context during execution of the program (see, for example, page 12, section 4.2, last paragraph, bulleted list, which shows encountering typing-context-relevant-code-points during execution);

(b) identifying a typing context handle associated with the typing context, the typing context handle referencing a typing context data structure associated with the typing context (see, for example, page 11, section 4.1, third paragraph, which shows identifying a unique identifier associated with the typing context, i.e. a typing context handle, that references a type descriptor, and page 12, section 4.2, first paragraph, which further shows that a type descriptor is a typing context data structure);

(c) computing the typing context data associated with the typing-context-relevant-code-point (see, for example, page 12, section 4.2, fourth paragraph, which shows creating a type descriptor associated with a typing-context-relevant-code-point, and page 12, section 4.2, first paragraph, which further shows that a type descriptor comprises typing context data);

(d) dynamically allocating a field in the typing context data structure associated with the typing-context-relevant-code-point, the field describing the exact type of the typing-context-relevant-code-point in the typing context (see, for example, page 11, section 4.1, third paragraph, which shows dynamically allocating a type descriptor that describes an exact type such as "Cell<Integer>," and page 12, section 4.2, first paragraph, which further shows that the type descriptor includes a field that stores a representation of the exact type); and

(e) recording the typing context data in the field of the typing context data structure (see, for example, page 11, section 4.1, second and third paragraphs, which shows recording the typing context data in the fields of the type descriptor).

With respect to claim 2 (original), the rejection of claim 1 is incorporated, and Viroli further discloses the limitations wherein the typing-context-relevant-code-point executes a type test on an instance of a generic class (see, for example, page 12, section 4.2, second paragraph, which shows instance tests), the typing context data includes a resource type descriptor defining the exact type of the instance (see, for example, page 13, Figure 9, which shows a type descriptor as defined in class "$TD," and page 11, section 4.1, second and third paragraphs, which shows an array of type descriptors and the exact type of the instance), and the computer process further comprises: performing the type test based on the resource type descriptor associated with the typing-context-relevant-code-point (see, for example, page 14, Figure 10, which shows examples of performing instance tests based on the type descriptor).

With respect to claim 3 (original), the rejection of claim 1 is incorporated, and Viroli further discloses the limitations wherein the typing-context-relevant-code-point executes an allocation of an instance of a generic class (see, for example, page 11, section 4.1, first paragraph, which shows instantiating a parametric type), the typing context data includes a resource type descriptor defining the exact type of the instance (see, for example, page 13, Figure 9, which shows a type descriptor as defined in class "$TD," and page 11, section 4.1, second and third paragraphs, which shows an array of type descriptors and the exact type of the instance), and the computer process further comprises: creating the instance of the generic class based on the resource type descriptor associated with the typing-context-relevant-code-point, wherein the instance is of the exact type (see, for example, page 11, section 4.1, first three paragraphs, which shows creating an instance based on the type descriptor, and page 12, section 4.2, last paragraph, second bullet).

With respect to claim 4 (original), the rejection of claim 1 is incorporated, and Viroli further discloses the limitations wherein the typing-context-relevant-code-point calls a generic method, the typing context data includes another typing context handle, and the computer process further comprises: passing the other typing context handle referencing the typing context data to the generic method as a hidden parameter (see, for example, page 12, section 4.1, first paragraph, which shows calling a method for friend types).

With respect to claim 5 (original), the rejection of claim 1 is incorporated, and Viroli further discloses the limitation wherein the identifying operation comprises: retrieving the typing context handle from a stack frame (see, for example, the abstract, which shows a Java virtual machine that implements a stack).

With respect to claim 7 (original), the rejection of claim 1 is incorporated, and Viroli further discloses the limitation wherein the computing operation comprises: retrieving the typing context data associated with the typing-context-relevant-code-point from a global hash table (see, for example, page 11, section 4.1, third paragraph, which shows retrieving the type descriptor from a hash table).

With respect to claim 8 (original), the rejection of claim 1 is incorporated, and Viroli further discloses the limitation wherein the encountering operation comprises: assigning an index to the typing-context-relevant-code-point (see, for example, page 11, section 4.1, third paragraph, which shows assigning a unique identifier).

With respect to claim 9 (original), the rejection of claim 8 is incorporated, and Viroli further discloses the limitation wherein the allocating operation comprises: allocating the field in the typing context data structure, in accordance with the index (see, for example, page 11, section 4.1, third paragraph, which shows dynamically allocating a type descriptor in accordance with a unique identifier).

With respect to claim 25 (previously presented), the claim is directed to an execution engine comprising subject matter that corresponds to subject matter recited in claim 1 (see the rejection of claim 1 above).

With respect to claim 26 (previously presented), the claim is directed to a method comprising subject matter that corresponds to the subject matter recited in claim 1 (see the rejection of claim 1 above).

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

9. Claims 6 and 10-14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Viroli, as applied to claim 1 above, in view of U.S. Pat. No. 5,093,914 to Coplien et al. (art of record, "Coplien").

With respect to claim 6 (original), the rejection of claim 1 is incorporated, and Viroli further discloses the limitation wherein the typing-context-relevant-code-point is executed within an instance of a generic class (see, for example, page 11, section 4.1, first paragraph, which shows instantiating a parametric type). Viroli does not expressly disclose the limitation wherein the identifying operation comprises:

(a) retrieving a first pointer to the instance; and

(b) retrieving the typing context handle via a second pointer, a second pointer being relative to the first point and referencing the typing context handle associated with the instance.

However, Coplien teaches step (a) above in terms of a first pointer to a window instance (see, for example, column 10, lines 14-21). Coplien further teaches step (b) above in terms of second pointer to a generic window (see, for example, column 10, lines 23-29) that is relative to the first pointer (see, for example, column 10, lines 31-35) and references type information associated with the instance (see, for example, column 10, lines 37-47).

Coplien is directed to executing parameterized polymorphic code (see, for example, column 10, lines 48-65). It would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teachings of Coplien into Viroli so as to include first and second pointers. The motivation for doing so would have been to facilitate the sharing of base or generic code by specific type instances or subclasses deriving from the same generic or base class or object, thus eliminating redundant code in subclasses which prefer the default semantics defined in the base class.

With respect to claim 10 (original), the rejection of claim 8 is incorporated. Viroli does not expressly disclose the limitation wherein the index is assigned based on the "arity" of the typing-context-relevant-code-point.

However, Coplien discloses this limitation in terms of an index based on the number of arguments (see, for example, column 12, lines 35-42).

Coplien is directed to executing parameterized polymorphic code (see, for example, column 10, lines 48-65). It would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teachings of Coplien into Viroli so as to assign the index based on "arity." The motivation for doing so would have been to facilitate the construction of function signatures (each consists of a function name and the number of arguments), for the purpose of organizing and identifying like-named functions that have different argument lists.

With respect to claim 11 (original), the rejection of claim 8 is incorporated. Viroli does not expressly disclose the limitation wherein the index is assigned based on a category associated with the typing-context-relevant-code-point.

However, Coplien discloses this limitation in terms of an index based on a category such as the number of arguments (see, for example, column 12, lines 35-42).

Coplien is directed to executing parameterized polymorphic code (see, for example, column 10, lines 48-65). It would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teachings of Coplien into Viroli so as to assign the index based on a category. The motivation for doing so would have been to facilitate the construction of function signatures (each consists of a function name and the number of arguments), for the purpose of organizing and identifying like-named functions that have different argument lists.

With respect to claim 12 (original), the rejection of claim 11 is incorporated, and Coplien further teaches the limitation wherein the category is assigned on a per-containing class basis (see, for example, FIG. 7, which shows an "XWindow" object and a "SunviewWindow" object).

With respect to claim 13 (original), the rejection of claim 11 is incorporated, and Coplien further teaches the limitation wherein the category is assigned on a per-containing method basis (see, for example, FIG. 7, which shows an "XWindow::draw" method and a "SunviewWindow::draw" method).

With respect to claim 14 (original), the rejection of claim 11 is incorporated, and Coplien further teaches the limitation wherein the category is assigned on a per-containing assembly basis (See, for example, FIG. 7, which shows an "XWindow" implementation and a "SunviewWindow" implementation).

Conclusion

10. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Yigdall whose telephone number is (571) 272-3707. The examiner can normally be reached on Monday through Friday from 7:30am to 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

MY

Michael J. Yigdall
Examiner
Art Unit 2192

mjy

[illegible]

TUAN DAM
SUPERVISORY PATENT EXAMINER